HEWLETT PACKARD

# A Light-Weight Application Sharing Infrastructure for Graphics Intensive Applications

Ming C. Hao, Dongman Lee*, Joseph S. Sventek
Software Technology Laboratory
HPL-96-114
July, 1996

concurrent engineering, application sharing, reduced event set, graphic intensive

We describe a light weight application sharing infrastructure that enables collaborative design using graphics intensive applications over low bandwidth networks. The basis of the technology employs an event-driven mechanism to share a reduced event set among multiple copies of an application executing on different workstations. This technology is referred to as RES-AP (Reduced Event Set Application Sharing). RES-AP allows geographically-dispersed engineers to work together on large complex problems with fast responses. This capability is achieved without modification to the applications or to the window system software.

Internal Accession Date Only

*Hewlett-Packard Workstation Systems Division, Technical Computing Center, Corvallis Laboratory.
To be published in and presented at *The Fifth IEEE International Symposium on High Performance Distributed Computing (HPDC-5)*, Syracuse, New York, August 6-9, 1996.

# A Light-Weight Application Sharing Infrastructure for Graphics Intensive Applications

**Ming C. Hao, Dongman Lee, Joseph S. Sventek**
**mhao@hpl.hp.com, dlee@cv.hp.com, sventek@hpl.hp.com**
**Hewlett-Packard, Palo Alto, CA**

## Abstract

We describe a light weight application sharing infrastructure that enables collaborative design using graphics intensive applications over low bandwidth networks. The basis of the technology employs an event driven mechanism to share a reduced event set among multiple copies of an application executing on different workstations. This technology is referred as RES-AP (Reduced Event Set Application Sharing). RES-AP allows geographically-dispersed engineers to work together on large complex problems with fast responses. This capability is achieved without modification to the applications or to the window system software.

## 1.0 Introduction

Technical enterprises in the 90's continuously face market pressures for higher quality products at a lower cost and in less time. To achieve this challenging goal, they have restructured the product development process to promote maximum parallelism - concurrent engineering [1]. All product related groups work together from the beginning and all information is shared, this is commonly termed virtual enterprise or virtual co-location.

Collaboration is one of the key technologies to enable a virtual enterprise, where people transparently access and share their information and knowledge without any location/organizational boundary. Numerous sharing technologies [2, 3, 4] based on X window systems have been proposed to fulfill user collaboration needs supporting application sharing without incurring any modification of existing application, data, and underlying computing environments.

These sharing technologies work by sharing X protocol requests with all the collaboration participants' X servers. However, due to protocol request replication, users often experience considerable performance degradation when X protocol requests from an application carry lots of data (e.g. bitmap transfer as a rendering mechanism), especially over low-bandwidth networks.

This paper describes a light weight application sharing infrastructure that enables collaboration among multiple users over low bandwidth networks. The paper is organized as follows. Chapter 2 provides an architectural overview of the existing works: the centralized architecture and replicated architecture. Chapter 3 describes how RES-AP approach differs from them. Chapter 4 provides detail description of RES-AP architecture.

## 2.0 Centralized vs. Replicated Application Sharing

Application sharing technologies are based on two basic architectures [4, 5, 6]: centralized and replicated application sharing mechanisms.

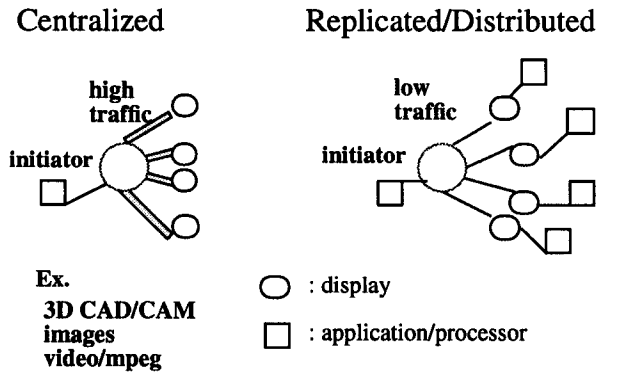
In the centralized architecture, there is only one instance of the shared application. All inputs to the application are sent to the execution site while the application's outputs (i.e. X protocol requests) are sent to all the displays, such as SharedX [3] and JVTOS' Shared Window System [2]. The centralized architecture provides an identical view by transmitting the X protocol to each user's display. Existing applications do not have to be modified, recompiled, or relinked.

The replicated architecture requires each participant to run locally his/her own copy of the shared application. Inputs are multicasted to each participant's shared application while outputs from the application are deliv-

ered only to its corresponding local display. The replicated architecture is light-weight but complex to synchronize due to the different process speeds. Examples are MMConf [7] and VConf [4]. Both require applications to be collaboration-aware.

The key differences between the two architecture are illustrated in Figure 1.

**FIGURE 1. Centralized vs. Replicated**




## 3.0 Our Approach

Our approach, RES-AP, is based on the replicated architecture. To achieve fast responses, RES-AP replicates application program data and execution on each site. For easy synchronization, RES-AP uses a floor control mechanism to guarantee that a single deterministic event stream is sent to the shared applications for processing.

RES-AP only shares a reduced set of events generated from user interactions instead of multiplexing all window messages (input/output, queries/ replies, and errors) among shared applications. There is no need to send X protocol requests and other unrelated events to shared workstations. Our technology resolves serious latency problems due to insufficient bandwidth. This enables graphics intensive application (e.g. 3D CAD/CAM) users to have an interactive, collaborative design session without significant performance degradation.

This approach assumes that if an identical set of applications are started in the same initial state and process an identical sequence of event sets, then the applications will continue to be in an identical state after processing each event.

The goal of the concurrent engineering collaboration research at the Hewlett-Packard Research Laboratories is to solve the following problems:

- light weight (work well over a typical spectrum of network bandwidths)
- no change to existing application and window systems
- support real-time 3D rendering (X based and non-X based rendering)

## 4.0 The RES-AP Architecture

RES-AP is built on a multiple server-clients model. It employs an event driven approach with a reduced event set. It contains three basic components:

1. event-driven and reduction (capturing)
2. event multicasting
3. event synchronization

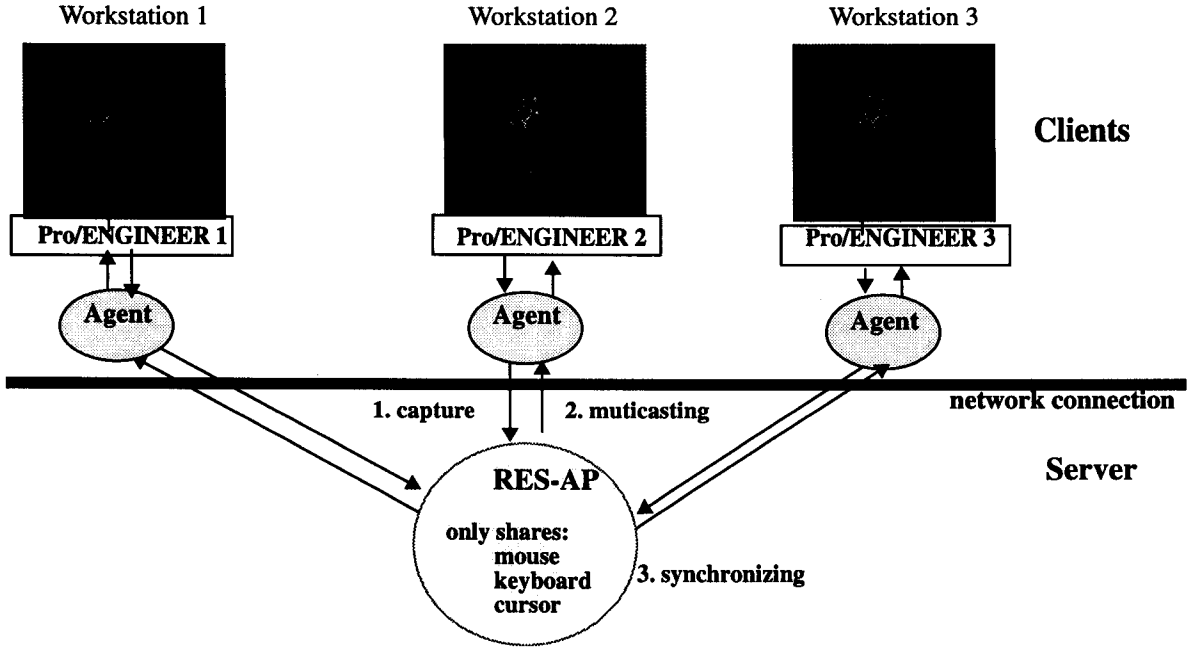
These components have the following characteristics:

- forces a single input event stream for easy synchronization through an explicit floor control mechanism.
- separates application program data, execution, and control for fast response time.

First, the RES-AP agent captures a reduced set of input events, such as mouse and keyboard inputs, and sends them to RES-AP. It analyzes the events and their states, puts them in proper execution order, and multicasts them to each instance of the shared application. As needed, RES-AP slows down the event processing to synchronize multiple views.

RES-AP employs an agent on each remote workstation to: (1) retrieve and send the local application window hierarchy array for mapping; (2) capture input events; (3) maintain event consistency among instances of the shared application; and (4) mediate environmen-

**FIGURE 2. RES-AP Architecture**




tal differences to allow for different window sizes, key codes, colormaps, etc.

Figure 2 illustrates the RES-AP overall architecture. Each of these components is described further in the following sections.

## 4.1 Event Driven & Reduction

A common method for sharing a window among multiple workstations by the same application is to intercept existing application messages and pass them to each user's workstation. With a centralized architecture, it usually generates heavy network traffic due to continual shipping of graphics primitives among the participating workstations.

RES-AP explores a new event capturing mechanism [8] instead of processing all the window messages. RES-AP captures only input events on a shared window that should be shared (e.g. user inputs) or synchronized (e.g. map notification). RES-AP ignores all other events.

## 4.2 Event Multicasting

After analyzing the captured input events, RES-AP [6] orders or groups them if necessary and sends the shared input events to the target application windows. Applications automatically trigger their own event handlers to execute received events. Events are processed just as they would be if the window events had been directly entered into the application windows.

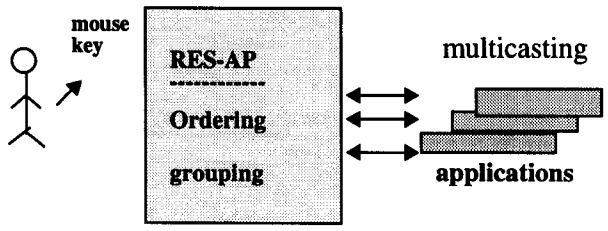
The key functions are:

- Grouping: to allow users to select the input event distribution scope.
- Ordering: to sequence events from multiple sources into a proper execution order.
- Multicasting: to distribute events to the appropriate targets.

To ensure the process is light-weight, RES-AP only multicast mouse, keyboard events/states, and cursor movements.

Figure 3 illustrates the RES-AP basic muticasting function flow.

**FIGURE 3.** Multicast User Events




## 4.3 Event concurrency control and synchronization

With a replicated execution approach, RES-AP synchronizes multiple copies of a shared application running on each site. The need for synchronization arises due to the mismatch of processing speeds of various processors. Our synchronization mechanism employs motion event compression and a two-phase protocol. RES-AP ensures that all shared applications are in a consist state before processing the next incoming event. In case synchronization is lost, RES-AP allows users to easily re-synchronize the application in local mode.

## 4.4 A Shared Cursor

RES-AP allows users to have a shadow image of the current floor holder's cursor; we call this a shared cursor mechanism. This improves visual perception of collaboration among participants since every cursor movement of the current floor holder's cursor is exactly replicated to the rest of the participants.

In order to synchronize the graphical output of shared applications on all displays, RES-AP defines two basic coordination methods with a light-weight reduced event set [9]:

- cursor movement compression to reduce network traffic.
- forced multicast of the last cursor movement to synchronize position of all participants' cursors before a non-cursor input event is sent, such as button press.

## 5.0 Summary

RES-AP is an on-going experiment in Hewlett-Packard Research Laboratories. We have built a series of increasingly powerful prototypes to demonstrate and evaluate the key technology in the X Window system. Our experience with the sharing of unmodified 3D CAD/CAM applications has been promising. The response time is almost instantaneous.

The most interesting feature of RES-AP is that we are able to share graphics intensive applications by capturing, multicasting, and synchronizing a reduced event set. This technology is extremely light weight (tested on as low as 56 kbps network). RES-AP has been used to support collaborative CAD/CAM 3D modeling among multiple workstations. Its event-multicast design center permits it to be usable over a variety of network bandwidths.

## Acknowledgment

Thanks to Mary Loomis and Chris Hsiung from HP Labs for their encouragement and suggestions, and to Jerrie Andreas, Milon Mackey for many technical suggestions and discussions.

## References

[1] Donald E. Carter, and Barbara Stiwell Baker, Concurrent Engineering: The Product Development Environment for the 1990s, Addison-Wesley Publishing Co., 1992.

[2] Thomas Gutekunst *et al,* "A Distributed and Policy-Free General Purpose Shared Window System," IEEE/ACM Transactions on Networking, Vol. 3, No. 1, Feb. 1995, pp. 52-62.

[3] John R. Portherfield, "Mixed Blessings" and "HP SharedX," HP Professional, Vol. 5, Issue 9, Sep, 1991.

[4] J. Chris Lauwers, Thomas A. Joseph, and Keith A. Lantz, "Replicated Architectures for Shared Window Systems: A Critique, "Proc. ACM Conf. Office Information System, 1990, pp. 249-260.

[5] S. R. Ahuja, J. R. Ensor, and S. E. Lucco "A Comparison of Application Sharing Mechanisms in Real-Time Desktop Conferencing Systems," Proc. ACM Conf. Office Information System 1990, pp. 238-248.

[6] Daniel Garfinkel, and Randy Branson, "A Comparison of Application Sharing Architectures in the X Environment," Xhibition 91.

[7]Terrence Crowley, Paul Milazzo, Ellie Baker, Harry Forsdick, and Raymond Tomlinson, "MMConf: An Infrastructure for Building Shared Multimedia Applications," Proc. ACM CSCW 90, Los Angeles, pp. 329-342.

[8] M. Hao, U. S. Patent pending on "A Mechanism to Control and Use Window Events Among Applications in Concurrent Computing" and "A Mechanism to Sense and Multicast to a Plurality of Exisiting Applications for Concurrent Execution in a Distributed Environment," Hewlett-Packard, Oct, 1994 and May, 1993.

[9] M. Hao, D. Lee, and J. Sventek, "A mechanism to share cursor for concurrent execution and consistent graphical views among plurality of existing applications," in preparation for US Patent.